high, the gesture **100** is considered to be associated with the button **66**. For example, if the overlap area **104** is more than 40% of the total area of either the bounding box **102** or the bounding box **88**, the gesture **100** can be considered to be associated with the button **66**.

The described stylus gestures, objects, and processes are preferably accomplished within the context of a "view system". In such a view system, various "views" or "objects" are stacked on top of each other, like pages of paper on a desk top. These views include a root view (such as the notepad) and virtually any number of views (within the limitations of the system) stacked on top of the root view.

The view system is a software routine which returns two pieces of information when the screen is engaged ("tapped") with a stylus. A first piece of information returned is which view or "object" was tapped. The second piece of information returned is the position of the tap on the tapped view. This location information is often returned in the form of Cartesian (x-y) coordinates. The view system therefore handles much of the routine input work for the computer system. Taps by stylus on non-active areas of the screen can be ignored by the view system. Likewise, inappropriate inputs on active areas of the screen can be ignored or can generate error conditions which can be acted upon by the system.

The term "object" has been used extensively in the preceding discussions. As is well known to software developers, an "object" is a logical software unit comprising data and processes which give it capabilities and attributes. For example, an object can be queried as to its type and can return such data as the number of words that it contains. Objects can contain other objects of the same or of a different type. Objects can also be used to project images on a screen according to their object type. There are many well known texts which describe object oriented programming. See, for example, *Object Oriented Programming for the Macintosh,* by Kurt J. Schmucher, Hayden Book Company, 1986.

In the present invention, objects are preferably implemented as part of a frame system that comprises frame objects related by a semantic network. A description of semantic networks can be found in "A Fundamental Tradeoff in Knowledge Representation and Reasoning", *Readings in Knowledge Representation,* by Brachman and Leveseque, Morgan Kaufman, San Mateo, 1985.

The use of object oriented programming, frame systems, and the aforementioned view system simplifies the implementation of the processes of the present invention. In FIG. 7A, a conceptual representation of various objects in view system is shown. The notepad application on the screen **42** forms a first or "root" layer, and the status bar **56** is positioned in a second layer "over" the root layer **42**. The clock **58** and buttons **60-70** are positioned in a third layer "over" the status bar **56**.

In FIG. 7*b*, a cross-section taken along line 7*b*—7*b* of FIG. 7*a* further illustrates the conceptual layering of various objects. The aforementioned viewing system automatically handles "taps" and other gestures of the stylus **38** on the screen **42** by returning information concerning which object has been gestured and where on the object the gesture occurred. For example, a gesture A on the screen **42** could create an action for the notepad application. A gesture B on the status bar **56** could be of part of a drag operation to move the status bar on the screen **42**. A gesture C on recognize button **66** can activate a process associated with that button. It is therefore clear that the object oriented programming and view system software makes the implementation of the processes of the present invention less cumbersome than traditional programing techniques.

In FIG. **8**, a process **200** for providing a gesture sensitive button for graphical user interface is illustrated. The process begins at step **202**, and, in a step **204**, it is determined whether the stylus **38** is on the screen **42**. If not, step **204** goes into a wait state. If the stylus is on the screen, data points are collected in a step **206** and it is determined in a step **208** whether the stylus has been lifted from the screen. If not, process control is returned to step **206**. After the stylus has been lifted from the screen, it is determined in a step **210** whether the collection of data points forms a gesture associated with a button in a step **210**. If the gesture is not associated with a button, the gesture is processed in a step **212** and the process **200** is completed as indicated at step **214**. If it was determined by step **210** that a gesture was associated with the button, then in a step **216** it is determined whether the gesture is a tap. If the gesture was a tap, the standard function for the button is performed in step **218** and the process is completed as indicated by step **214**. If the gesture is associated with a button but is not a tap, then a step **220** determines whether the gesture is relevant to the button. If the gesture is not relevant (i.e. that gesture means nothing to that button) then the process is completed as indicated at **214**. If the gesture is relevant to the button, then an alternative button action is processed in step **222** and the process is completed at step **214**.

In FIG. **9**, a script table **224** helps determine: (a) whether the gesture is relevant to the button; and (b) what to do when that gesture is detected in association with the button. In this example, a button detects three different types of gestures, a "tap" **226**, a "check-mark" **228**, and an "X-mark" **230**. If the tap gesture **226** is detected, the "script" for the indicated process is to: (1) highlight the button **66** momentarily; (2) toggle (i.e. turn off if on, turn on if off) the recognizers; and (3) reverse the button state. As described previously, the button **66** indicates that the recognizers are on when the button says "recognize", and indicates that they are turned off when there is a diagonal line through the word "recognize". When the check-mark gesture **228** is found to be associated with the button, the process script is to: (1) highlight the button momentarily; and (2) pop up the recognizer palette **78**. When the X-mark gesture **230** is detected, the script is: (1) highlight the button **66** momentarily; (2) turn on all of the recognizers; and (3) show the recognizer button in the "on" state. In this example, other gestures performed on the button **66** are considered "non-relevant". Also, a tap, check-mark, or X-mark gesture performed elsewhere on the screen **42** would not be considered relevant to the button **66**.

It is desirable that a given gesture should initiate a similar type of process regardless of which button it contacts. For example, a check mark could always means "START", an X-mark could always mean "STOP", etc. The button then provides the specific context for the command initiated by the gesture.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. A gesture sensitive button for a graphical user interface comprising: